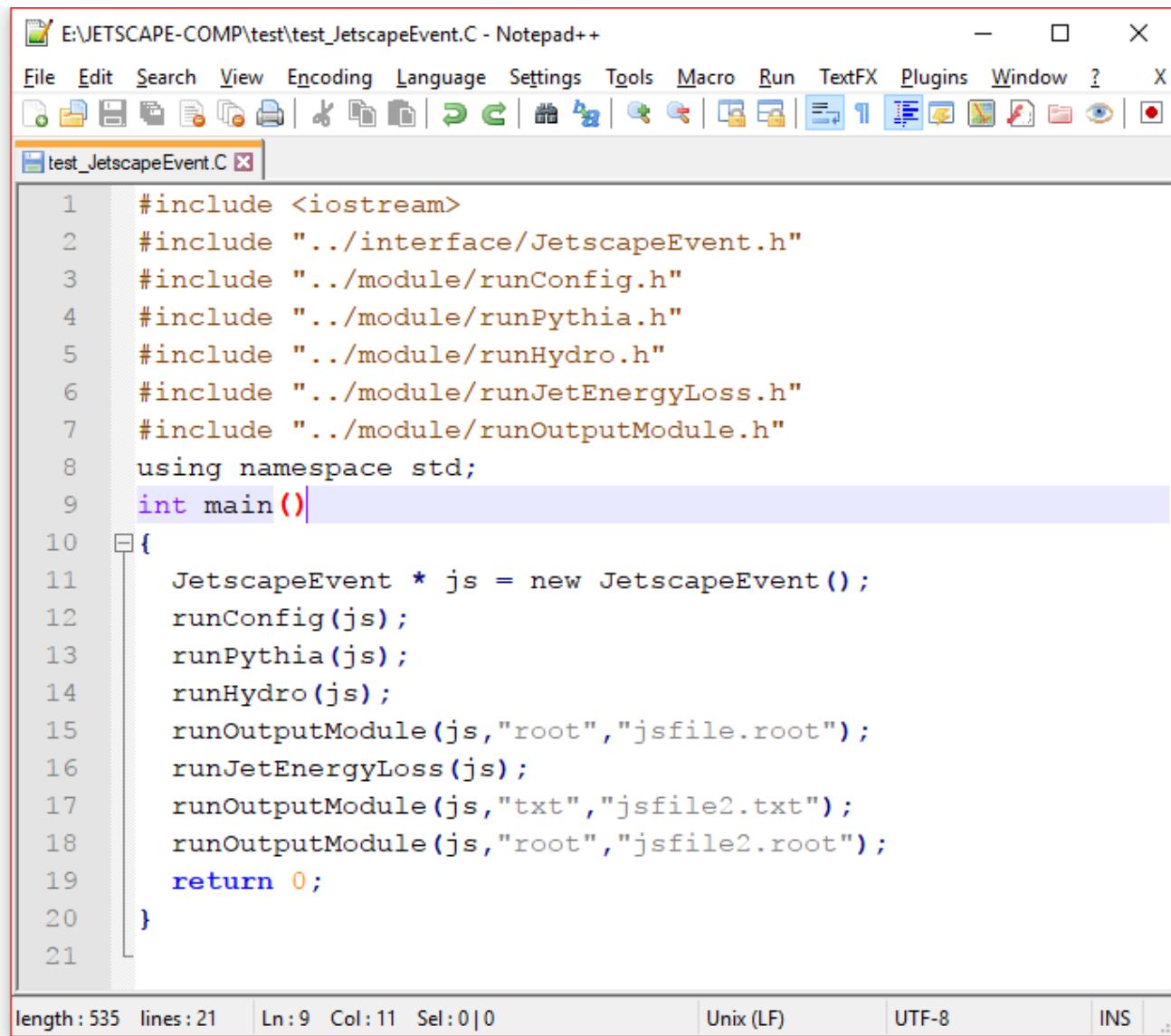


Framework Demo

Demo overview

- JetscapeEvent class data structure (config, jets, and partons)
- Example (working) code which creates one event and different modules interact with it
 - Using Pythia as an external package to generate jets
 - Using brick hydro code I found in test directory
 - A dummy jet energy loss class
 - Input / output modules
 - Root integration example

Top level view of main



The screenshot shows a Notepad++ window with the file `E:\JETSCAPE-COMP\test\test_JetscapeEvent.C` open. The code is a C++ program that includes several header files and defines a `main` function. The code is as follows:

```
1 #include <iostream>
2 #include "../interface/JetscapeEvent.h"
3 #include "../module/runConfig.h"
4 #include "../module/runPythia.h"
5 #include "../module/runHydro.h"
6 #include "../module/runJetEnergyLoss.h"
7 #include "../module/runOutputModule.h"
8 using namespace std;
9 int main()
10 {
11     JetscapeEvent * js = new JetscapeEvent();
12     runConfig(js);
13     runPythia(js);
14     runHydro(js);
15     runOutputModule(js,"root","jsfile.root");
16     runJetEnergyLoss(js);
17     runOutputModule(js,"txt","jsfile2.txt");
18     runOutputModule(js,"root","jsfile2.root");
19     return 0;
20 }
```

The Notepad++ interface includes a toolbar with various icons for file operations, a status bar at the bottom showing file length, line count, and character count, and a menu bar with options like File, Edit, Search, View, Encoding, Language, Settings, Tools, Macro, Run, TextFX, Plugins, Window, and Help.

Top level view of main

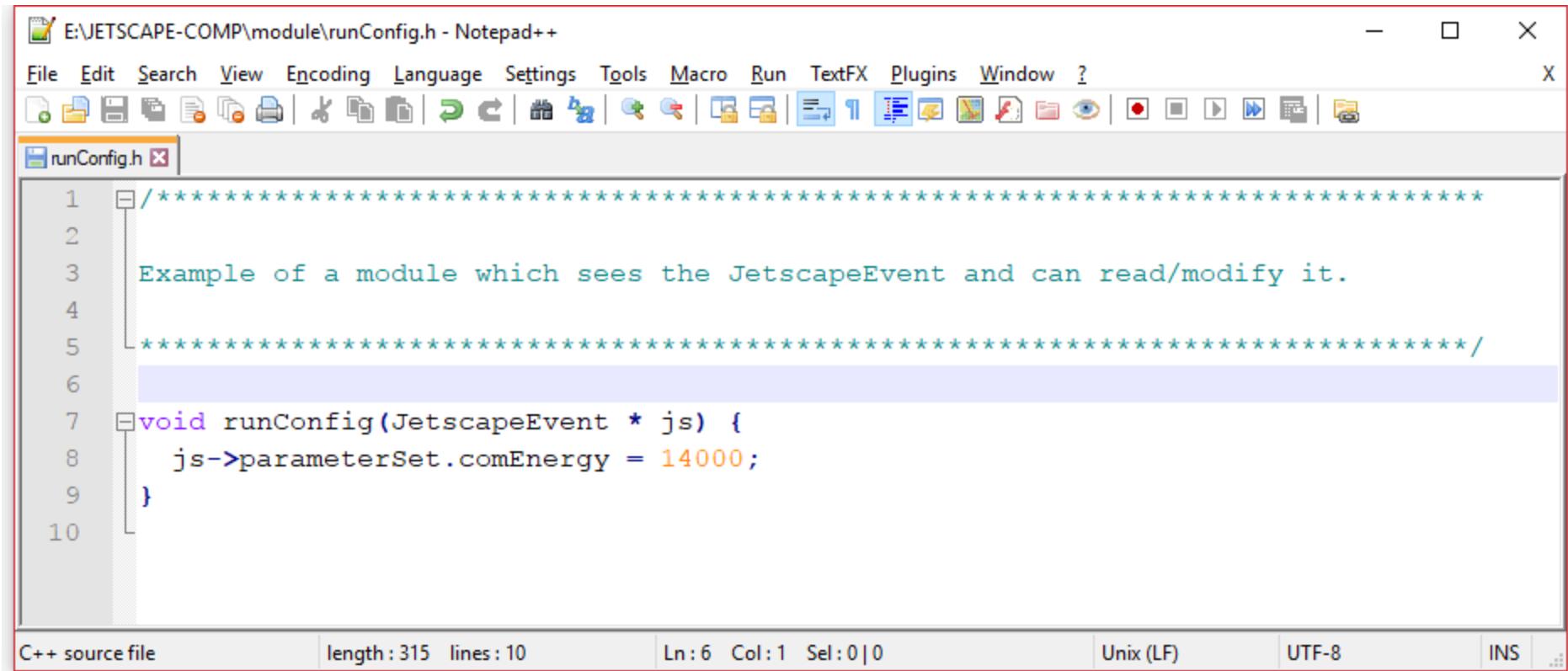
```
E:\JETSCAPE-COMP\test\test_JetscapeEvent.C - Notepad++
File Edit Search View Encoding Language Settings Tools Macro Run TextFX Plugins Window ?
test_JetscapeEvent.C x

1 #include <iostream>
2 #include "../interface/JetscapeEvent.h"
3 #include "../module/runConfig.h"
4 #include "../module/runPythia.h"
5 #include "../module/runHydro.h"
6 #include "../module/runJetEnergyLoss.h"
7 #include "../module/runOutputModule.h"
8 using namespace std;
9 int main()
10 {
11     JetscapeEvent * js = new JetscapeEvent();
12     runConfig(js);
13     runPythia(js);
14     runHydro(js);
15     runOutputModule(js,"root","jsfile.root");
16     runJetEnergyLoss(js);
17     runOutputModule(js,"txt","jsfile2.txt");
18     runOutputModule(js,"root","jsfile2.root");
19     return 0;
20 }
```

length : 535 lines : 21 Ln : 9 Col : 11 Sel : 0 | 0 Unix (LF) UTF-8 INS

- Load JetscapeEvent data structure class
- Load modules
- Load outputmodule
- Create event instance
- Run modules that read and modify JetscapeEvent
- Write event to file before running JetEnergy Loss
- Run jet energy loss
- Write out event after jet modifications, root and txt file

Example module: runConfig



The screenshot shows the Notepad++ interface with the file `E:\JETSCAPE-COMP\module\runConfig.h` open. The code in the editor is:

```
1 // ****
2
3 Example of a module which sees the JetscapeEvent and can read/modify it.
4
5 ****
6
7 void runConfig(JetscapeEvent * js) {
8     js->parameterSet.comEnergy = 14000;
9 }
10
```

The status bar at the bottom provides file statistics: length: 315, lines: 10, Ln: 6 Col: 1 Sel: 0|0, and encoding information: Unix (LF), UTF-8, INS.

Example module: runPythia - 1/2

EJETSCAPE-COMP\module\runPythia.h - Notepad++

File Edit Search View Encoding Language Settings Tools Macro Run TextFX Plugins Window ?

runPythia.h

```
1 // **** Example of a module which sees the JetscapeEvent and can read/modify it.
2
3 ****
4
5 #include <string>
6 #include "Pythia8/Pythia.h"
7 using namespace Pythia8;
8
9
10
11 void runPythia(JetscapeEvent * js) {
12
13     // Number of events, generated and listed ones.
14     int nEvent      = 1;
15     int nListJets = 5;
16
17
18     // Generator. LHC process and output selection. Initialization.
19     Pythia pythia;
20     char numstr[21]; // enough to hold all numbers up to 64-bits
21     sprintf(numstr, "%d", js->parameterSet.comEnergy);
22     std::string comE = "Beams:eCM = ";
23     comE += numstr;
24     comE += ".";
25     pythia.readString(comE);
26     pythia.readString("HardQCD:all = on");
27     pythia.readString("PhaseSpace:pTHatMin = 200.");
28     pythia.readString("Next:numberShowInfo = 0");
29     pythia.readString("Next:numberShowProcess = 0");
30     pythia.readString("Next:numberShowEvent = 0");
31     pythia.init();
32 }
```

C++ source file length : 2,463 lines : 79 Ln : 8 Col : 19 Sel : 0 | 0 Unix (LF) UTF-8 INS

<https://gist.github.com/velicanu/f7985dabb6379b313d99b04720eb87f5>

Is passed JetscapeEvent pointer

Read comEnergy from JetscapeEvent , give it to Pythia

Example module: runPythia – 2/2

EJETSCAPE-COMP\module\runPythia.h - Notepad++

```
File Edit Search View Encoding Language Settings Tools Macro Run TextFX Plugins Window ?  
runPythia.h  
47 // Set up CellJet jet finder.  
48 CellJet cellJet( etaMax, nEta, nPhi, nSel);  
49  
50 // Histograms. Note similarity in names, even when the two jet finders  
51 // do not calculate identically the same property (pT vs. ET, y vs. eta).  
52 Hist nJetsD("number of jets, CellJet - SlowJet", 45, -22.5, 22.5);  
53  
54 // Begin event loop. Generate event. Skip if error.  
55 for (int iEvent = 0; iEvent < nEvent; ++iEvent) {  
56     if (!pythia.next()) continue;  
57  
58     // Analyze Slowjet jet properties. List first few.  
59     slowJet.analyze(pythia.event);  
60     if (iEvent < nListJets) slowJet.list();  
61  
62     // Fill SlowJet inclusive jet distributions.  
63     for (int i = 0; i < slowJet.sizeJet(); ++i) {  
64         js->jetCollection.push_back(Jet(0,slowJet.pT(i),slowJet.y(i),slowJet.phi(i)  
65         ));  
66     }  
67  
68     // Compare number of jets for the two finders.  
69     nJetsD.fill( cellJet.size() - slowJet.sizeJet() );  
70  
71     // End of event loop. Statistics. Histograms.  
72 }  
73 pythia.stat();  
74 cout << nJetsD ;  
75 }  
76
```

length : 2,423 lines : 76 Ln : 76 Col : 1 Sel : 0 | 0 Unix (LF) UTF-8 INS

<https://gist.github.com/velicanu/f7985dabb6379b313d99b04720eb87f5>

Loop through jets
Pythia found and add
them to JetscapeEvent
jet collection - [link](#)

Example module: runHydro

```
// Written by Chun Shen
#include <iostream>
#include <cstring>

#include "fluid_dynamics.h"
#include "brick_jetescape.h"

void runHydro(JetscapeEvent * js) {
    Parameter parameter_list;

    parameter_list.hydro_input_filename = "";

    Brick *brick_ptr = new Brick();
    brick_ptr->initialize_hydro(parameter_list);
    brick_ptr->evolve_hydro();
    FluidCellInfo* check_fluid_info_ptr = new FluidCellInfo;
    brick_ptr->get_hydro_info(1.0, 0.0, 0.0, 0.0, check_fluid_info_ptr);
    std::cout<<"\033[1;31mprinting hydro\033[0m"<<std::endl;
    brick_ptr->print_fluid_cell_information(check_fluid_info_ptr);

    std::cout<<"\033[1;31mbefore jet energy loss, reading jets in hydro
code\033[0m"<<std::endl;
    for (size_t i = 0; i < js->jetCollection.size(); i++) {
        std::cout<<"found jet: "<<i<<" "<<js->jetCollection[i].pt<<" "<<js->jetCollection[i].eta<<" "<<js->jetCollection[i].phi<<" "<<std::endl;
    }
}
```

Is passed JetscapeEvent pointer

Here this hydro module has full info about the jets we created in the Pythia module

Example module: runJetEnergyLoss

```
#include <cstdlib>
#include <iostream>

void runJetEnergyLoss(JetscapeEvent * js) {
    //read randomNumberSeed from JetscapeEvent
    srand (static_cast<unsigned> (js->parameterSet.randomNumberSeed) );
    for (size_t i = 0; i < js->jetCollection.size(); i++) {
        //randomly multiply each jet by by number from 0 to 1 for "energy loss"
        js->jetCollection[i].pt *= static_cast<float> (rand()) / static_cast<float> (RAND_MAX);
    }
    std::cout<<"\033[1;33m after jet energy loss, reading jets in jet energy loss
    code\033[0m"<<std::endl;
    for (size_t i = 0; i < js->jetCollection.size(); i++) {
        std::cout<<"found jet: "<<i<<" "<<js->jetCollection[i].pt<<" "<<js->jetCollection[i].eta<<" "<<js->jetCollection[i].phi<<" "<<std::endl;
    }
}
```

Is passed JetscapeEvent pointer

Here each jet pT is randomly scaled between 0-1, random number seed set from config file

Example module: runOutputModule

```
1 #include <iostream>
2 #include <fstream>
3 #include "TFile.h"
4 #include "TTree.h"
5
6 void txtOutputModule(JetscapeEvent * js, std::string filename) {
7
8 void rootOutputModule(JetscapeEvent * js, std::string filename) {
9
10 void runOutputModule(JetscapeEvent * js, std::string type, std::string filename) {
11     if(type.compare("txt")==0 || type.empty()) txtOutputModule(js,filename);
12     else if(type.compare("root")==0) rootOutputModule(js,filename);
13     else std::cout<<"Warning: output type not supported, no output written."<<std::endl;
14 }
```

C++ source file | length : 2,108 lines : 58 | Ln : 58 Col : 1 Sel : 0 | 0 Unix (LF) UTF-8 INS

Is passed JetscapeEvent pointer

Implementation is hidden, but here we can write out the JetscapeEvent to either a plain text file, or a root file in TTree form

Let's run

E:\JETSCAPE-COMP\test\test_JetscapeEvent.C - Notepad++

File Edit Search View Encoding Language Settings Tools Macro Run TextFX Plugins Window ?

test_JetscapeEvent.C

```
1 #include <iostream>
2 #include "../interface/JetscapeEvent.h"
3 #include "../module/runConfig.h"
4 #include "../module/runPythia.h"
5 #include "../module/runHydro.h"
6 #include "../module/runJetEnergyLoss.h"
7 #include "../module/runOutputModule.h"
8 using namespace std;
9 int main()
10 {
11     JetscapeEvent * js = new JetscapeEvent();
12     runConfig(js);
13     runPythia(js);
14     runHydro(js);
15     runOutputModule(js,"root","jsfile.root");
16     runJetEnergyLoss(js);
17     runOutputModule(js,"txt","jsfile2.txt");
18     runOutputModule(js,"root","jsfile2.root");
19     return 0;
20 }
```

C source file length : 535 lines : 21 Ln : 21 Col : 1 Sel : 0 | 0 Unix (LF) UTF-8 INS ...

Demo part 2:

```
E:\JETSCAPE-COMP\test\test_readJS.C - Notepad++
File Edit Search View Encoding Language Settings Tools Macro Run TextFX Plugins Window ?
test_readJS.C x
1 #include <iostream>
2 #include "../interface/JetscapeEvent.h"
3 #include "../module/runJetEnergyLoss.h"
4 #include "../module/runOutputModule.h"
5 #include "../module/runInputModule.h"
6
7 using namespace std;
8
9 int main(int argc, char *argv[])
10 {
11     JetscapeEvent * js = runInputModule("root","jsfile.root");
12     runJetEnergyLoss(js);
13     runOutputModule(js,"txt","j_file3.txt");
14     runOutputModule(js,"root","jsfile3.root");
15     return 0;
16 }
17
```

C source file | length : 422 lines : 17 | Ln : 17 Col : 1 Sel : 0 | 0 Unix (LF) UTF-8 INS

Here we read in the JetscapeEvent from the file we wrote out in the first demo. No need to run Pythia or hydro again!

Here we could test out different jet energy loss models on the same input.

For this example we're going to run the same to show we get the same result as in the first demo.

Let's run demo part 2

E:\JETSCAPE-COMP\test\test_readJS.C - Notepad++

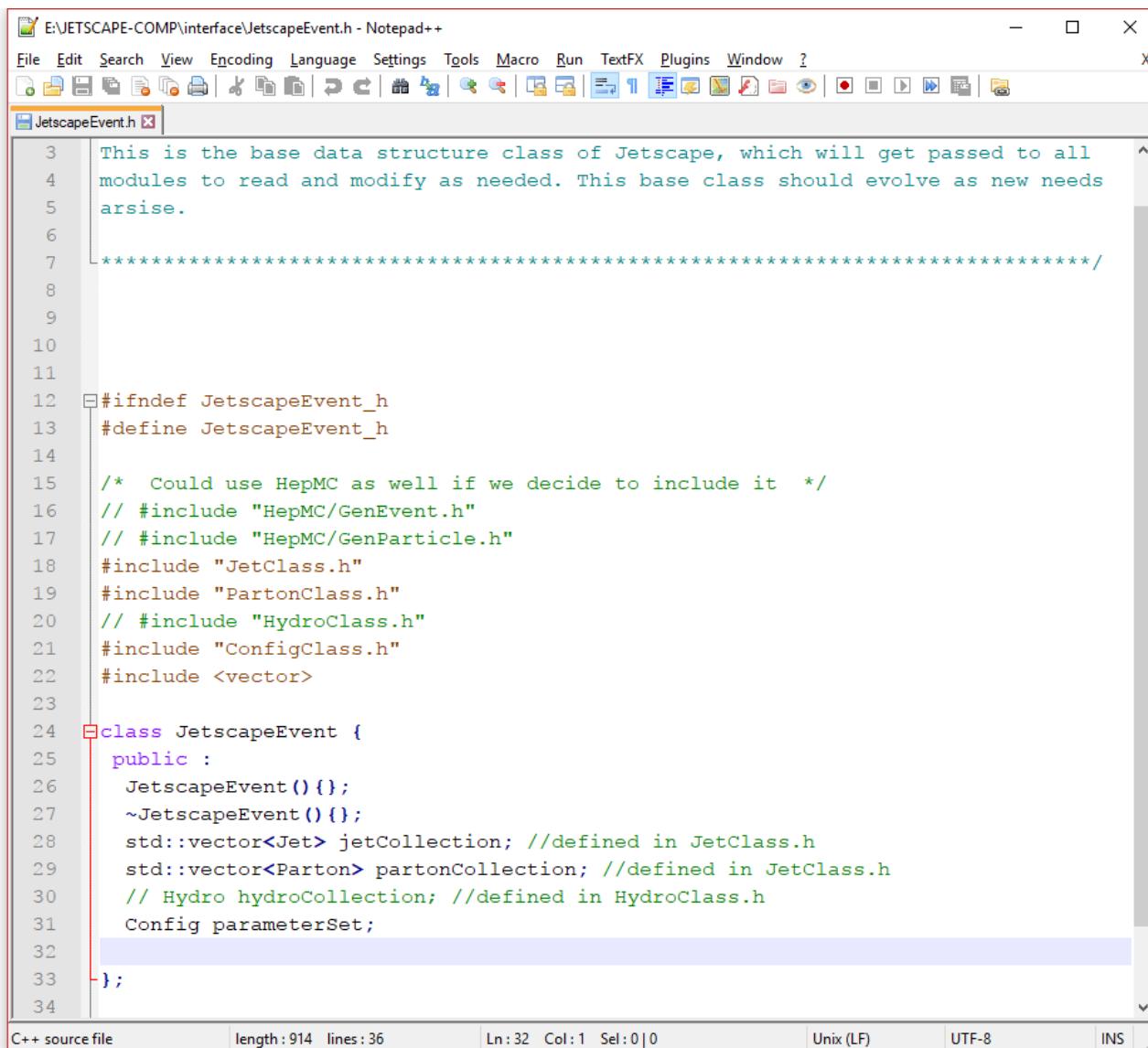
File Edit Search View Encoding Language Settings Tools Macro Run TextFX Plugins Window ?

test_readJS.C

```
1 #include <iostream>
2 #include "../interface/JetscapeEvent.h"
3 #include "../module/runJetEnergyLoss.h"
4 #include "../module/runOutputModule.h"
5 #include "../module/runInputModule.h"
6
7 using namespace std;
8
9 int main(int argc, char *argv[])
10 {
11     JetscapeEvent * js = runInputModule("root","jsfile.root");
12     runJetEnergyLoss(js);
13     runOutputModule(js,"txt","jsfile3.txt");
14     runOutputModule(js,"root","jsfile3.root");
15     return 0;
16 }
17
```

C source file length : 422 lines : 17 Ln : 17 Col : 1 Sel : 0 | 0 Unix (LF) UTF-8 INS ...

Review what is JetscapeEvent



The screenshot shows a Notepad++ window displaying the content of the file `JetscapeEvent.h`. The code defines a base class for Jetscape events, including declarations for jet and parton collections and a configuration parameter set.

```
3 This is the base data structure class of Jetscape, which will get passed to all
4 modules to read and modify as needed. This base class should evolve as new needs
5 arsise.
6
7 ****
8
9
10
11
12 #ifndef JetscapeEvent_h
13 #define JetscapeEvent_h
14
15 /* Could use HepMC as well if we decide to include it */
16 // #include "HepMC/GenEvent.h"
17 // #include "HepMC/GenParticle.h"
18 #include "JetClass.h"
19 #include "PartonClass.h"
20 // #include "HydroClass.h"
21 #include "ConfigClass.h"
22 #include <vector>
23
24 class JetscapeEvent {
25 public :
26     JetscapeEvent(){};
27     ~JetscapeEvent(){};
28     std::vector<Jet> jetCollection; //defined in JetClass.h
29     std::vector<Parton> partonCollection; //defined in JetClass.h
30     // Hydro hydroCollection; //defined in HydroClass.h
31     Config parameterSet;
32
33 };
34
```

At the bottom of the window, status bar information includes: C++ source file, length: 914 lines: 36, Ln: 32 Col: 1 Sel: 0|0, Unix (LF), UTF-8, INS.

Summary

- We ran a “hello world” framework
 - Different modules interact with the JetscapeEvent
 - Can add more modules the same way
 - Can extend JetscapeEvent to have more useful info
 - Can replace my demo Jet, Parton, etc. classes with what’s already written
- Wrote out some output
- Can run on the output we write out and write out modified output (modular!)

DIY

```
git clone git@github.com:velicanu/JETSCAPE-COMP.git
cd JETSCAPE-COMP/
git checkout demo_2017_03_10
#add Pythia, only needed for runPythia part of example
cd external_packages
wget
http://home.thep.lu.se/~torbjorn/pythia8/pythia8223.tgz
tar -xvf pythia8223.tgz
mv pythia8223 pythia
cd pythia
./configure
make # making pythia here
cd ../..
make # making jetscape test macros here
./test/test_JetscapeEvent.exe
./test/test_readJS.exe
```

<https://gist.github.com/velicanu/19e54c2610df5ae2a5c7c4fa17c8d255>